Drop Lab

Lab 5

Section 7

Submitted By:

Benjamin Muslic

Submission Date:

10/26/2021

Lab problem

The two week long lab aims to further consolidate C problem solving skills with the implementation of functions and loops this time.The first part required the user to make a code to show the falling distance in meters of the DUALSHOCK 4 controller. The controller was dropped from the 2nd floor stairs at Coover hall and the goal was to obtain the most accurate result possible which was around 10 meters. The second part of the lab required using the same code and adding to it so it can calculate air resistance. The same drop process was done again.

Analysis

Through the implementation of lecture and ZyBooks exercises, the user is expected to make a c program that measures the distance (and air resistance) from the fall of the 2nd floor. What was required is to use of several loops and else statements as well as the magnitude function. The user is then expected to show results to the Ta for testing. The user also needs to alter his source code to add in air resistance Calculation. Finally the user is expected to write a lab report representing both parts.

Design

Source code #1 (Just the fall)

```c
#include <stdio.h>
#include <math.h>
#include <time.h>

#define TRUE 1
#define FALSE 0
#define UP 2
#define DOWN 3
#define LEFT 4
#define RIGHT 5
#define FRONT 6
#define BACK 7

double mag (double x,double y,double z);
int close_to(double tolerance, double point, double value);

int main() {
    int t, printed, timing, falling, k = 1000, last_print;
    double time_start, time_stop, ax, ay, az, msec;
    int b[6];

    printf("Benjamin Muslic\n");
    printf("muslicb@iastate.edu");
    printed = FALSE, falling == FALSE;
    last_print = 0;
    while(TRUE) {
        scanf("%d, %lf, %lf, %lf, %d, %d, %d, %d, %d, %d", &t, &ax, &ay, &az, &b[0], &b[1], &b[2], &b[3], &b[4], &b[5] );
        if(printed == FALSE) {
            printf("Atomic batteries to power, turbines to speed. Roger. Let's go!");
            printed = TRUE;
        }
        if(last_print == 0) {
            last_print = t;
        }
        if(close_to(3, last_print+k, t) == TRUE) {
            if(falling == FALSE) {
                printf(".");
            } else {
                printf("!");
            }
            last_print = t;
        }

        if((close_to(0.1, 2, ax) == TRUE || close_to(0.1, 2, ay) == TRUE || close_to(0.1, 2, az) == TRUE) && close_to(0.001, 1, mag(ax, ay, az)) == FALSE) {
            if(falling == FALSE) {
                printf("\n Holy levitation Batman! we are flying! ");
                time_start = t;
            }

            falling = TRUE;
        } else {
            if(falling == TRUE) {
            double diff;
            diff = (-1)*(time_stop - time_start)/1000.0;
            printf("\n Holy Hang Time Batman! We flew %lf meters in %lf seconds.", (double)(0.0 + (mag(ax,ay,az)*diff) + (1.0/2.0) * (-9.8) * diff * diff), diff);
            break;
            }
            time_stop = t;
            falling = FALSE;
        }



        fflush(stdout);
    }
    return 0;
}



double mag(double x,double y,double z) {
    return sqrt(pow(x,2) + pow(y,2) + pow(z,2));
}

int close_to(double tolerance, double point, double value) {
    int close;

    if(value > point - tolerance && value < point + tolerance) {
        close = TRUE;
    } else {
        close = FALSE;
    }

    return close;
}
```

With the combination of several functions and loops, the source code was able to measure falling distance in meters from the second floor.

Source code #2 (now with air resistance)

```c
/*-----------------------------------------------------------------
-                          SE/CprE 185 Lab 04
-             Developed for 185-Rursch by T.Tran and K.Wang
-----------------------------------------------------------------*/

/*-----------------------------------------------------------------
-                              Includes
-----------------------------------------------------------------*/
#include <stdio.h>
#include <math.h>
#include <time.h>


/*-----------------------------------------------------------------
-                              Defines
-----------------------------------------------------------------*/
#define FALSE 0
#define TRUE 1
#define UP 2
#define DOWN 3
#define LEFT 4
#define RIGHT 5
#define FRONT 6
#define BACK 7



/*-----------------------------------------------------------------
-                             Prototypes
-----------------------------------------------------------------*/
int closeto(double tol, double point, double val);
double mag(double ax, double ay, double az);

/*-----------------------------------------------------------------
-                           Implementation
-----------------------------------------------------------------*/
int main(void) {
    int t, printed, timing, falling, last;
    double tStart, tStop, ax, ay, az;

    printf("Benjamin Muslic\n");

    printed = FALSE;
    falling = FALSE;
    last = 0;
    printf("Atomic batteries to power, turbines to speed. Roger. Let's go!");
    sscanf("%d, %lf, %lf, %lf", &t, &ax, &ay, &az );

    while(closeto(0.1, 1, mag(ax, ay, az))) {
        sscanf("%d, %lf, %lf, %lf", &t, &ax, &ay, &az);
        if(last == 20) {
            printf(".");

            last = 0;
        }
        last++;
    fflush(stdout);
    }

    tStart = t;
    last = 0;
    int preT = t;
    double dist =0;
    double preVel = 0;
    printf("\n Holy levitation Batman! we are flying! ");

    while( mag(ax, ay, az)<2.5) {
    sscanf("%d, %lf, %lf, %lf", &t, &ax, &ay, &az);
    double acc = 9.8*(1-mag(ax,ay,az));
    double deltaT = (t - preT)/1000.0;
    preT = t;

    double vel = preVel + acc*deltaT;
    preVel = vel;

    dist = dist+ vel*deltaT;
    if(last == 20) {
            printf("!");
            fflush(stdout);
            last = 0;
        }
        last++;
    fflush(stdout);
    }
    t = t-tStart;
    double d = .5*(9.8)* pow((t/1000.0),2);
    printf("\n Holy Hang Time Batman! We flew %lf meters in %lf seconds.",d,t/1000.0);
    printf("\n Wind resistance is %lf meters which is %lf %% less" ,dist, 100 - (dist/d)*100);



return 0;
}


/* Put your functions here */
double mag(double ax, double ay, double az) {
    double x2 = pow(ax, 2);
    double y2 = pow(ay, 2);
    double z2 = pow(az, 2);

    return sqrt(x2 + y2 + z2);


    }

int closeto(double tol, double point, double val) {
    int close;

    if((val > point - tol) && (val < point + tol)) {
        close = TRUE;
    }
    else {
        close = FALSE;
    }
    return close;
}
```

Tinkered code with regards to air resistance. Changed a few variables to stop confusing myself. Overall it turned out pretty nice after changing it several times.

Testing

The testing is related to the questions the lab was asking. The testing is as follows.

Do the same drop 5 times in the classroom and record the distances. How consistent are your results? What could cause any variation?

```
muslicb@CO1318-08 /cygdrive/u/CPRE185/Lab5
$ ./ds4rd.exe -d 054c:09cc -D DS4_USB -t -g | ./lab5_5
Benjamin Muslic
Atomic batteries to power, turbines to speed. Roger. Let's go!.
 Holy levitation Batman! we are flying! !!
 Holy Hang Time Batman! We flew 2.007040 meters in 0.640000 seconds.
 Wind resistance is 0.455225 meters which is 77.318613 % less
```

```
muslicb@CO1318-08 /cygdrive/u/CPRE185/Lab5
$ ./ds4rd.exe -d 054c:09cc -D DS4_USB -t -g | ./lab5_5
Benjamin Muslic
Atomic batteries to power, turbines to speed. Roger. Let's go!.
 Holy levitation Batman! we are flying! !
 Holy Hang Time Batman! We flew 1.728896 meters in 0.594000 seconds.
 Wind resistance is 0.426177 meters which is 75.349777 % less
```

```
muslicb@CO1318-08 /cygdrive/u/CPRE185/Lab5
$ ./ds4rd.exe -d 054c:09cc -D DS4_USB -t -g | ./lab5_5
Benjamin Muslic
Atomic batteries to power, turbines to speed. Roger. Let's go!.
 Holy levitation Batman! we are flying! !
 Holy Hang Time Batman! We flew 1.914063 meters in 0.625000 seconds.
 Wind resistance is 0.477901 meters which is 75.032085 % less
```

```
muslicb@CO1318-08 /cygdrive/u/CPRE185/Lab5
$ ./ds4rd.exe -d 054c:09cc -D DS4_USB -t -g | ./lab5_5
Benjamin Muslic
Atomic batteries to power, turbines to speed. Roger. Let's go!
 Holy levitation Batman! we are flying! !
 Holy Hang Time Batman! We flew 1.817317 meters in 0.609000 seconds.
 Wind resistance is 0.388312 meters which is 78.632700 % less
```

```
muslicb@CO1318-08 /cygdrive/u/CPRE185/Lab5
$ ./ds4rd.exe -d 054c:09cc -D DS4_USB -t -g | ./lab5_5
Benjamin Muslic
Atomic batteries to power, turbines to speed. Roger. Let's go!
 Holy levitation Batman! we are flying! !
 Holy Hang Time Batman! We flew 1.547636 meters in 0.562000 seconds.
 Wind resistance is 0.455973 meters which is 70.537455 % less
```

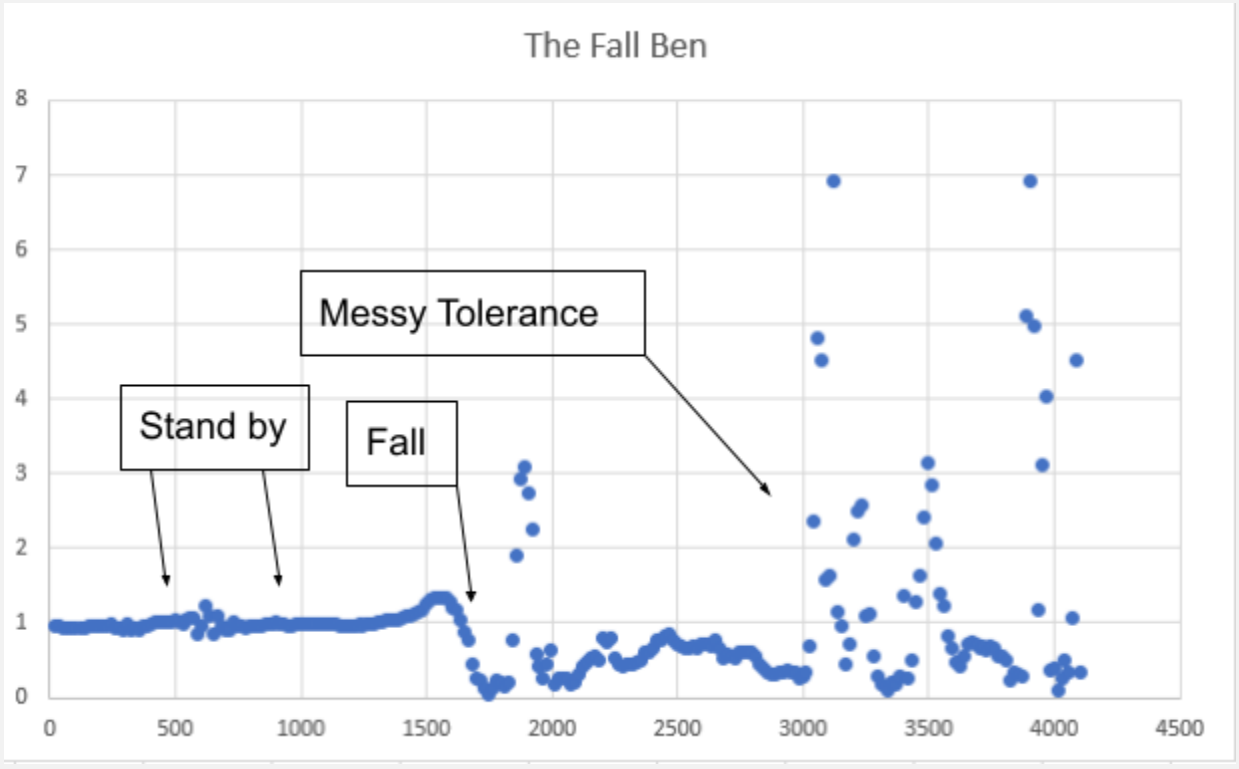
As shown in the screenshots above, after constant tinkering of the code, the results seem to be quite consistent. I know from experience that the primary cause of variation is the timing of the drop. One wrong move and the output would give a wild result (for ex 20+ meters fall). You have to smoothly drop the controller the same way to get consistent results.

How far is it from the third floor railing to the bottom floor according to your code, using the sample data?

```
muslicb@CO1318-08 /cygdrive/u/CPRE185/Lab5
$ ./lab5_5.exe <lab5_sampledata2013.csv
Benjamin Muslic
Atomic batteries to power, turbines to speed. Roger. Let's go!..
 Holy levitation Batman! we are flying! !!!!!
 Holy Hang Time Batman! We flew 11.350772 meters in 1.522000 seconds.
 Wind resistance is 8.887740 meters which is 21.699244 % less
muslicb@CO1318-08 /cygdrive/u/CPRE185/Lab5
```

According to the sample code, it seems as though the approximate fall would be around 11.5 meters. This seems pretty accurate.

In your report, include a graph of the magnitude of the acceleration as a function of time from the sample data. Label where the freefall is happening, where it hits the ground, and your tolerances. Explain and justify any tolerances you are using.



The screenshot above shows the graph of the fall in action. It starts off with me holding the box in the air until the TA shouts to drop. The dip is the fall and the impact is scattered everywhere most likely due to the messy tolerance or the controller bouncing around in the box.

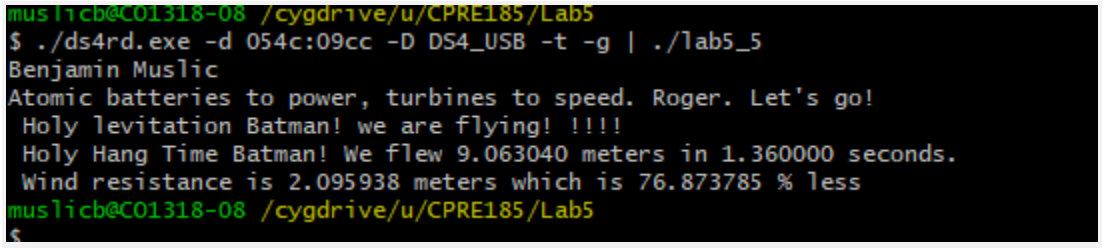
Part 2

Analysis (continued)

How much difference tends to occur in drops in the lab area? From the 2nd floor?

Mine was halfway decent. I had originally had problems with the magnitude function as my TA suggested. I had to fix this in order to obtain a more accurate result. With the help of lab mates, I was able to get my output to display pretty much the same throughout.

How far is it from the second floor railing to the bottom floor according to your code?



According to my code and also in comparison with other peoples drops, around 10 meters.

What issues arose in implementing Part 2?

The issue was trying to edit my decently sized code. There was several issues in which my code refused to compile. I had to implement several debugging steps to get my code to work. It is simple C problem solving.

Comments

This was a very challenging lab (especially during calculus exam) but probably the most interesting out of all of them (seriously). There was the coding experience and the overall dropping experience. It was nice to see something I personally developed be put into action. It will most likely be better from now.